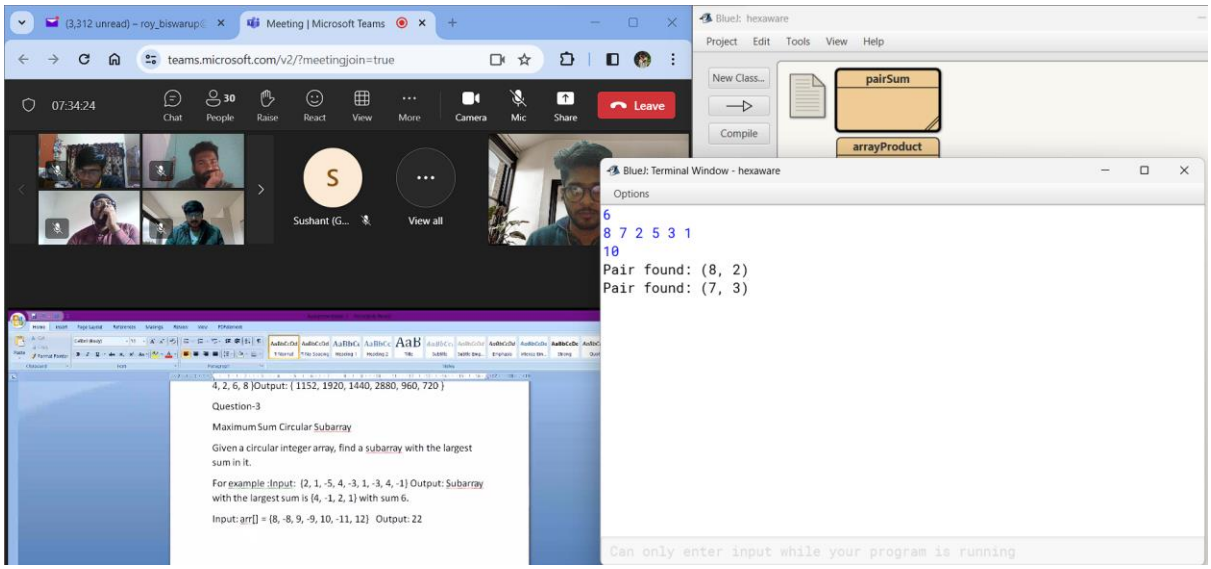
PROBLEM 1

```
import java.io.*;
import java.util.*;
class pairSum {
   public static void findGivenSumPair(int[] arr, int sum)
{
      HashMap<Integer, Integer> map = new
HashMap<>();
      int flag = 0;
      for (int i = 0; i < arr.length; i++) {
         int curr = arr[i];
         int diff = sum - curr;
         if (map.containsKey(diff) ) {
            System.out.println("Pair found: (" + diff + ", " +
curr + ")");
            flag=1;
         }
         map.put(curr, i);
      }
      if(flag == 0)
         System.out.println("Pair not found");
```

```
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int [] arr = new int [n];
        for(int i=0;i<n;i++){
            arr [i] = sc.nextInt();
        }
        int sum = sc.nextInt();
        findGivenSumPair(arr, sum);
    }
}
```

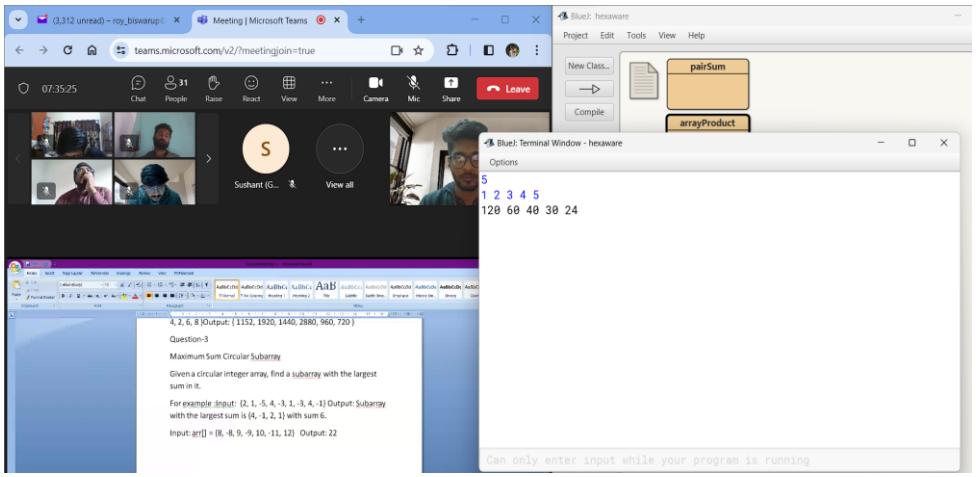


Output 1

PROBLEM 2

```
import java.io.*;
import java.util.*;
class arrayProduct{
   public static int[] getProductWithOtherElement(int[]
nums,int n) {
       int[] leftProducts = new int[n];
       int[] rightProducts = new int[n];
       int[] result = new int[n];
       int leftProduct = 1;
       for (int i = 0; i < n; i++) {
          leftProducts[i] = leftProduct;
          leftProduct *= nums[i];
       }
       int rightProduct = 1;
       for (int i = n - 1; i >= 0; i--) {
          rightProducts[i] = rightProduct;
          rightProduct *= nums[i];
       }
       for (int i = 0; i < n; i++) {
```

```
            result[i] = leftProducts[i] * rightProducts[i];
        }
        return result;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int [] nums = new int [n];
        for(int i=0;i<n;i++){
            nums [i] = sc.nextInt();
        }
        int[] result = getProductWithOtherElement(nums,n);
        for(int res:result)
            System.out.print(res+" ");
    }
}
```

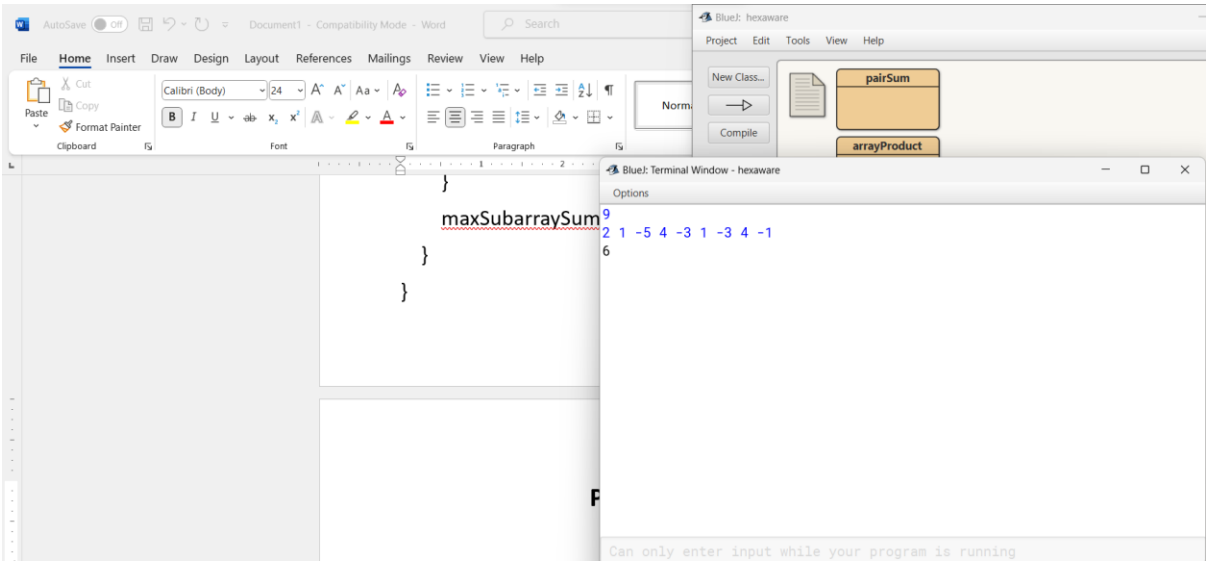


Output 2

PROBLEM 3

```
import java.io.*;
import java.util.*;
class largestSumSubarray{
    public static void maxSubarraySumCircular(int[]
arr,int n) {
        int maxSum = Integer.MIN_VALUE;
        // int start = 0;
        // int end = 0;
        for (int i = 0; i < n; i++) {
            int currentSum = 0;
            //int tempStart = i;
            for (int j = 0; j < n; j++) {
                int index = (i + j) % n;
                currentSum = Math.max(arr[index],
currentSum + arr[index]);
                /* if (currentSum == arr[index]) {
                    tempStart = index;
                } */
                if (currentSum > maxSum) {
                    maxSum = currentSum;
                    // start = tempStart;
```

```
                // end = index;
            }
        }
    }
    /* System.out.print("Subarray with the largest sum
is ");
    for (int i = start; i <= end; i++) {
        System.out.print(arr[i]+" ");
    }
    System.out.println("whose sum is "+ maxSum); */
    System.out.println(maxSum);
  }
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int n = sc.nextInt();
    int [] arr = new int [n];
    for(int i=0;i<n;i++){
        arr [i] = sc.nextInt();
    }
    maxSubarraySumCircular(arr,n);
  }
}
```

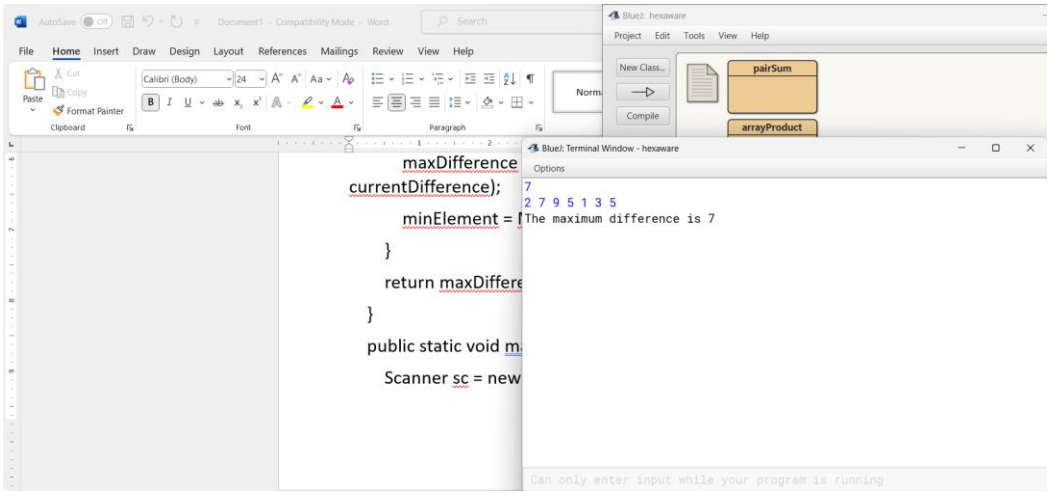

File Home Insert Draw Design Layout References Mailings Review View Help
Paste Cut Copy Format Painter
Clipboard Font Paragraph
BlueJ: hexaware
Project Edit Tools View Help
New Class...
Compile
pairSum
arrayProduct
}
maxSubarraySum
}
}
BlueJ: Terminal Window - hexaware
Options
9
2 1 -5 4 -3 1 -3 4 -1
6
Can only enter input while your program is running

Output 3

PROBLEM 4

```
import java.util.*;
class maxDifference{
    public static int findMaxDifferencePairs(int[] arr,int n)
{
        if (n < 2) {
            return -1;
        }
        int minElement = arr[0];
        int maxDifference = arr[1] - arr[0];
        for (int i = 1; i < n; i++) {
            int currentDifference = arr[i] - minElement;
            maxDifference = Math.max(maxDifference,
currentDifference);
            minElement = Math.min(minElement, arr[i]);
        }
        return maxDifference;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
```

```
        int n = sc.nextInt();
        int [] arr = new int [n];
        for(int i=0;i<n;i++){
            arr[i] = sc.nextInt();
        }
        int res = findMaxDifferencePairs(arr,n);
        if(res<0){
            System.out.println("Enter sufficient data");
        }else{
            System.out.println("The maximum difference is "
+ res);
        }
    }
}
```

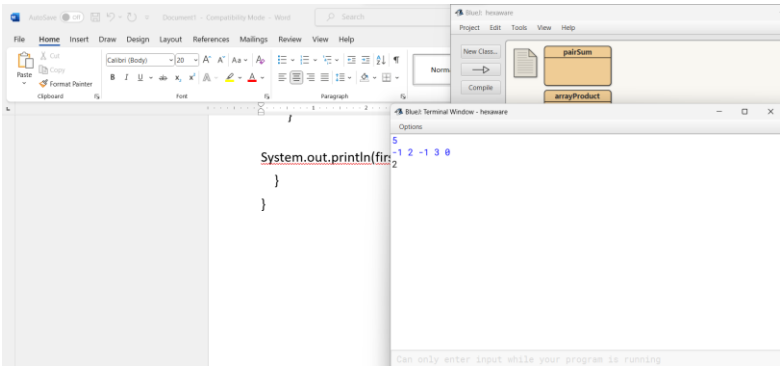


Output 4

PROBLEM 5

```
import java.util.*;
import java.io.*;
public class FirstNonRepeatingElement {
   public static int firstNonRepeatingElement(int[] nums,int n) {
      HashMap<Integer, Integer> frequencyMap = new HashMap<>();
      HashMap<Integer, Integer> indexMap = new HashMap<>();
      for (int i = 0; i < n; i++) {
         int num = nums[i];
         frequencyMap.put(num, frequencyMap.getOrDefault(num, 0) + 1);
         if (frequencyMap.get(num) == 1) {
            indexMap.put(num, i);
         }
      }
      int result = Integer.MAX_VALUE;
      for (int num : frequencyMap.keySet()) {
         if (frequencyMap.get(num) == 1) {
```

```
                result = Math.min(result,
indexMap.get(num));
            }
        }
        return (result == Integer.MAX_VALUE) ? -1 :
nums[result];
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int [] arr = new int [n];
        for(int i=0;i<n;i++){
            arr[i] = sc.nextInt();
        }
System.out.println(firstNonRepeatingElement(arr,n));
    }
}
```

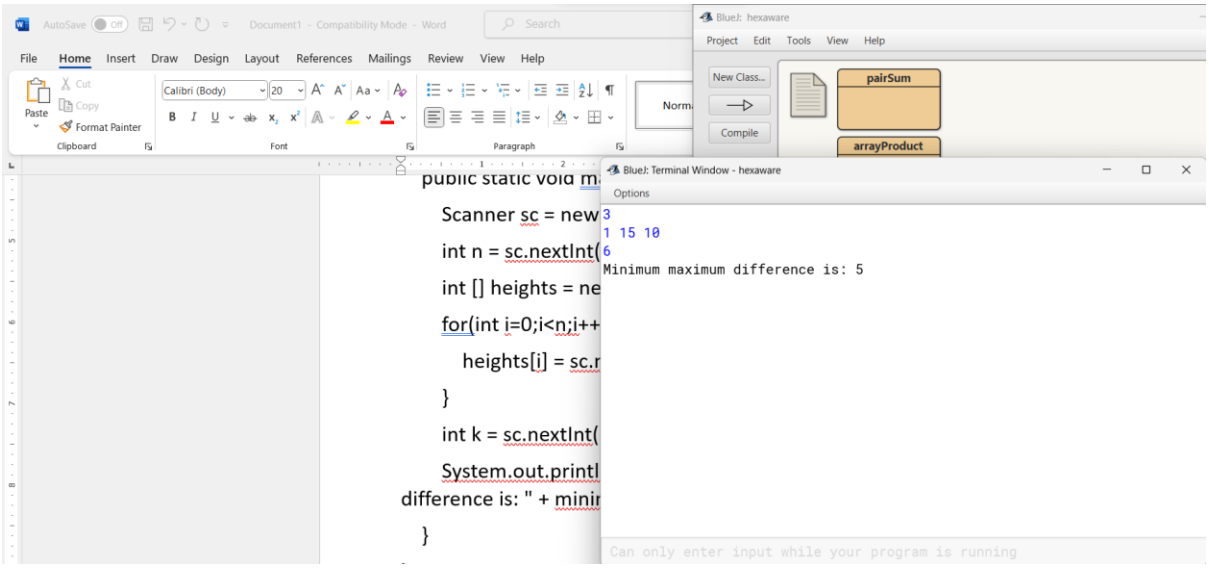


Output 5

PROBLEM 6

```
import java.util.*;
import java.io.*;
class MinimizeMaxDifference {
    public static int minimizeMaxDifference(int[] heights,
int k,int n) {
        Arrays.sort(heights);
        int result = heights[n - 1] - heights[0];
        int small = heights[0] + k;
        int big = heights[n - 1] - k;
        if (small > big) {
            int temp = small;
            small = big;
            big = temp;
        }
        for (int i = 1; i < n - 1; i++) {
            int subtract = heights[i] - k;
            int add = heights[i] + k;
            if (subtract >= small || add <= big) {
                continue;
```

```
            }
            if (big - subtract <= add - small) {
                small = subtract;
            } else {
                big = add;
            }
        }
        return Math.min(result, big - small);
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int [] heights = new int [n];
        for(int i=0;i<n;i++){
            heights[i] = sc.nextInt();
        }
        int k = sc.nextInt();
        System.out.println("Minimum maximum
difference is: " + minimizeMaxDifference(heights, k,n));
    }
}
```


BlueJ: hexaware
Project Edit Tools View Help
New Class...
Compile
pairSum
arrayProduct
BlueJ: Terminal Window - hexaware
Options
3
1 15 10
6
Minimum maximum difference is: 5
Can only enter input while your program is running
Scanner sc = new
int n = sc.nextInt(
int [] heights = ne
for(int i=0;i<n;i++
heights[i] = sc.r
}
int k = sc.nextInt(
System.out.printl
difference is: " + minir
}

OUTPUT 6